A Vision Based Aerial Robot solution for the Mission 7 of the International Aerial Robotics Competition

Jose Luis Sanchez-Lopez , Jesús Pestana , Jean-Franois Collumeau , Ramon Suarez-Fernandez , Pascual Campoy , and Martin Molina

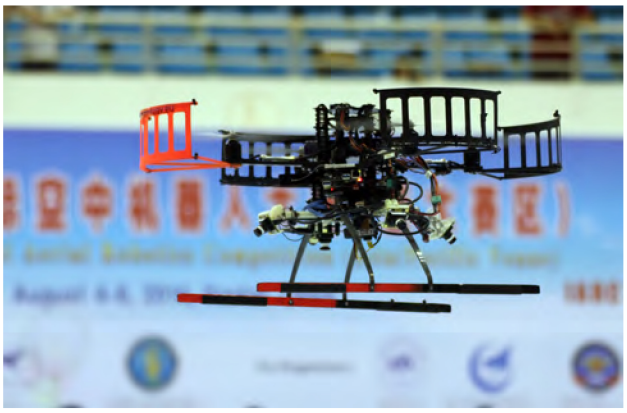


Fig. 1: Ascending Technologies Pelican Quadrotor with our onboard equipment in a competition flight in the China venue of the competition

I. Introduction

The International Aerial Robotics Competition (IARC) mission 7, aims for competitors to develop solutions for some of the major unsolved problems in UAV systems: GPS and Laser denied navigation, robot-robot interaction, and Obstacle avoidance. In order to test all of these challenges, an indoor arena was created where no static objects are within laser range. Inside this arena, 10 iRobot Create are programmed to navigate with a pseudo-random movement and 4 iRobots are moving in circles carrying poles as dynamic obstacles. The challenge of the IARC mission 7 is to locate and "herd" in less than 10 minutes at least seven of the iRobots toward one end of the arena. The UAV has to fully-autonomously interact with them, by touching them, while avoiding the dynamic obstacles.

In order to overcome the problem of aerial navigation in a GPS/Laser denied environment, we proposed a visual-based solution, as a continuation of the work presented in [10].

The Self-Localization is achieved by means of a visual odometry combined with a Simultaneous Localization And Mapping (SLAM) algorithm based on visual keypoints. The identification of targets and threats inside the arena are achieved by means of computer vision algorithms. Both provide information to an artificial intelligence module in charge of the tactical decisions needed to successfully complete the challenge that sends commands to the controllers. All these components will be presented in the section III.

We planed the IARC competition as a two-year project, aiming to complete both parts of the 7th mission by the 2015 edition. The milestones achieved by the 2014 competition were the following:

- Set-up the Aerial Vehicle, including not only the platform but also the onboard sensors, the computers and the communications. See section II for additional details;
- Develop the software framework inside which the system architecture is going to be developed;
- Develop a system architecture allowing the team to participate in the IARC while meeting the necessary safety requirements. This architecture includes the modules explained in section III;
- Develop the simulated environment for field testing, including the Arena, ground target robots and moving obstacles. See section IV;

We avoided for this year the detection and interaction with other aerial robots.

The system has been partially validated before the competition both in simulated environments and in a replica arena designed following the guidelines and informations available in the IARC mission 7-a rules, ensuring test scenario condi-

tions close to the competition's.

The mission was not completed by any participant in the 2014 edition, but we were awarded with two special prizes in the China Venue: Best Target Detection and Best System Control (see section IV-C).

II. Aerial Robot

In this section, a complete description of the aerial robot will be exposed.

A. Platform: AscTec Pelican

The AscTec Pelican Quadrotor made by Ascending Technologies is used as the main platform. The vehicle structure, motors, and rotors were used without modification, nevertheless, a new landing gear was fabricated along with camera and sensor mounts. The main reason for choosing this platform is that it has an structure convenient for prototyping. The main body is designed like a tower, which allows us to change anything quickly and easily. As a result of its high payload capacity and its modular design it is able lift all the sensors needed to solve this challenge.

The AscTec AutoPilot is a precise and multifunctional research tool mounted on the UAV. This flight control unit contains an IMU with other integrated sensors, two onboard ARM7 microprocessors, and various communication interfaces (see left block of figure 2).

We directly connect our onboard computer to the AscTec Autopilot through the Low-Level Processor, sending it Pitch, Roll, Thrust and speed of Yaw commands, and receiving the information given by its integrated sensors.

B. Onboard Hardware

1) Sensors: Only visual sensors were added to the quadrotor. Five uEye Cameras were installed, four of them around the perimeter of the quadrotor (front, back, right and left) and the last one in the bottom of the quadrotor. One optical flow sensor was also installed in the bottom of the quadrotor. See figure 1.

The uEye cameras, the UI-1221LE, are fitted with the Aptina color WVGA sensor. The global shutter enables you to capture fast-moving objects. Thanks to the optional HDR (high dynamic range) mode, the CMOS sensor delivers excellent images even in high dynamic, high contrast scenes. The 1/3" sensor format makes the camera the ideal replacement for standard analog video cameras at no great expense.

The PX4Flow is an optical flow smart camera (it provides the image for setup purposes, but it not designed to capture images). It has a native resolution of 752 x 480 pixels and calculates optical flow on a 4x binned and cropped area at 250 Hz (bright, outdoors), giving it a very high light sensitivy. Unlike many mouse sensors, it also works indoors and in low outdoor light conditions without the need for an illumination LED at 120 Hz (dark, indoors). It also counts with an ultrasound sensor to measure the flying altitude. The output of the sensor is the ground speed plus the flying altitude.

2) Computer: Since all the processes are going to be done on-board, a powerful and lightweight computer is needed. The On-board computer chosen for this task is the Mastermind by Ascending Technologies. This computer has a powerful Intel Core i7-3612QE (4 x 2.1 GHz) with 4 GB DDR3 RAM in addition to a huge variety of standard interfaces to connect the onboard sensors (1x FireWire, 5x USB 2.0 and 2x USB 3.0).

C. Communications and Ground Computer

The Onboard Computer runs the software architecture developed by our team (see section III) and is connected to the Asctec Autopilot through a UART cabled connection. The communication with the ground station is achieved by means of a 5GHz WiFi link from the Asctec Mastermind. The ground station is only used for user monitoring purposes and to start the mission execution. The Onboard sensors are connected to the Onboard Computer through USB ports. See figure 2 for a better understanding of the communication links.

D. Safety Features

For overall safety during flights certain steps have been taken. First, there exist a safety pilot that can take manual control of the UAV if needed, using a 2.4 GHz dedicated communication channel that interacts directly with the AscTec Autopilot, overwriting all control commands sent by the onboard computer. Second, the quadrotor has been equipped with propeller protection which could be the most harmful part of the vehicle. Third, the landing gear has been equipped with padding as to protect the on-board sensors, equipment and the environment in case of malfunction or emergency landing. Automatic shutoff of the propellers and system has also been implemented in two ways: The first is software related where an emergency shutoff state can be called, this state shuts the power off to all the motor bringing it to a dead stop. The second is a kill switch that interrupts current to the motors by means of electronics; this signal to shut the power off is sent through a private communication channel to the aerial vehicle. See layer 0 and layer 1 of figure 4 for more details on safety features.

III. System Architecture

The proposed system architecture has six main modules as can be seen in figure 3: The **Aerial Vehicle** block has been described in section II. The **Localization and Mapping** module (section III-A) is in charge of the environment perception by the drone and the pose and map estimation. The **Mission Planner** (section III-D) decides iteratively the list of actions that the drone has to execute during the mission, monitoring the state of the drone and the map in real-time. The **Controller** module (section III-B) allows the drone to achieve the atomic actions (take-off, hover, move, land, touch target robot) given by the Mission Planner. The **Supervisor** (section III-C) has a triple mission in the system: it monitors and controls the state of all the modules, it communicates with the HMI, and it decides the control mode

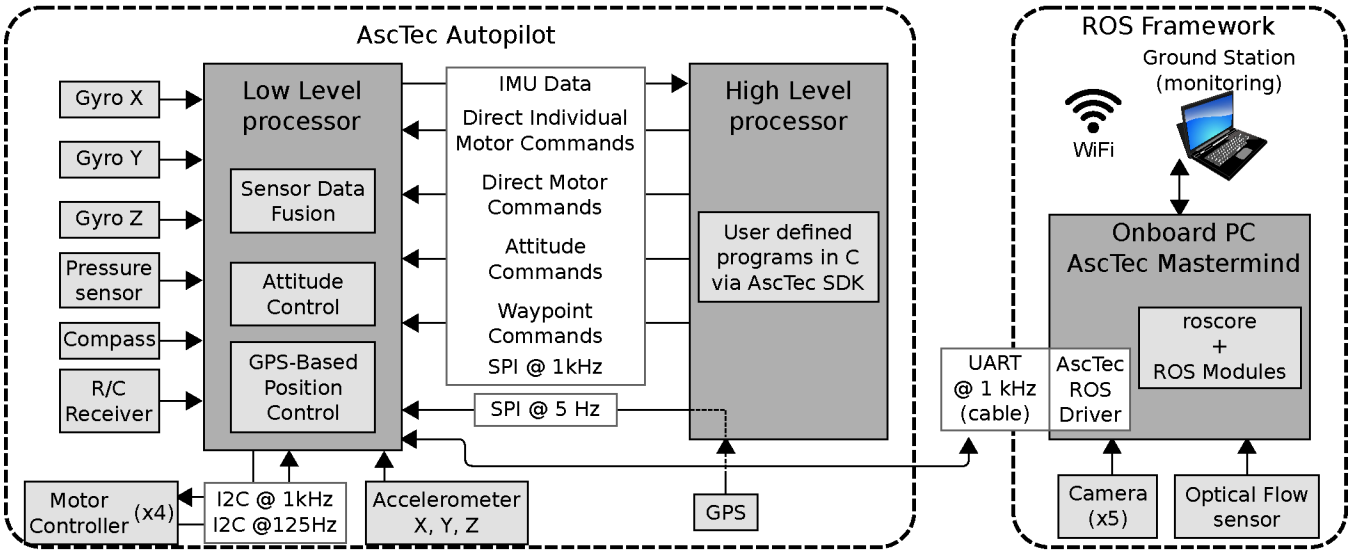


Fig. 2: Main communications diagram of the team's aerial robot architecture. Safety communication has been omitted for clarity

taking into account if the actions are feasible. The **HMI** (section III-E) allows the operators to monitor the state of the drone and send it commands.

This system architecture is inspired in the following previous works of the authors: [13] and [8].

A risk workflow diagram is presented in Figure 4 that summarizes the required dependencies of the main modules and modes of navigation. This risk workflow diagram divides the architecture in layers to understand the functioning requirements related to the various modules of the architecture. Layer 0, as required by the competition rules, a kill-switch can disconnect the vehicle from the battery. Layer 1, the safety pilot can override the whole software architecture by pressing a switch on the radio controller. Layer 2, the multirotor driver has several flying modes to ease testing and mission execution (taking-off, hovering, flying, landing and emergency). Also, the visual servoing controller requires the proper functioning of a visual tracker, but does not use the ground optical flow measurements. Layer 3, the rest of the architecture requires the proper execution of the computer vision algorithms and the localization and mapping modules. Layer 4, The System Supervisor interfaces the Mission Planner with the rest of the architecture; and it is able to change the operating mode of the architecture to move the robot away from map borders and obstacles. Layer 5 is the whole architecture.

A. Localization and Mapping

The Localization and Mapping module is in charge of localizing the drone in the Arena, as well as finding and identifying Target Ground Robots and Moving Obstacles, with enough precision and speed to allow the drone to complete the mission.

It is composed of the following four main modules:

- Odometry State Estimator (section III-A.1), which fuses the measurements received from some of the onboard drone sensors (IMU, Optical Flow and Altitude) to calculate an estimate of the state of the drone, using its module and a standard Extended Kalman Filter (EKF);
- Computer Vision Algorithms (section III-A.2), which process the pictures taken by the cameras to extract high-level features that can be used easily;
- Drone Localization and Arena Mapping (section III-A.3), which takes the grid landmarks given by the CV Module and the estimated pose of the drone given by the Odometry State Estimator to create a local map of the Arena and to locate the drone in this map. Using the information of the key lines (colored borders of the grid) given by the CV Module and the previous local map, the drone can be located in the Arena;
- Target Ground Robots and Moving Obstacles Localization (section III-A.4), which estimates the pose and location of the target ground robots and moving obstacles using the information given by the CV Module.

1) Odometry-based State Estimation: The Odometry-based State Estimator is based on an EKF algorithm (see [14]) that fuses the control inputs of the drone and the odometry measurements (IMU + Optical Flow + Altitude) to obtain an estimate of the pose and the speed of the drone. Further details are shown in [12], [11].

2) Computer Vision Algorithms: The information provided by computer vision means is of two natures. First, visual landmarks need to be provided for the mapping of the arena and the localization of the UAV inside it. Target- and threat-related information is also needed in order to successfully achieve the competition's objectives.

The general appearance of pictures taken in our test arena is as follows: a grid of homogeneous lines characterized by a high luminance value are drawn on a non-uniform background in which appearance may be locally similar to the grid lines. Possible light reflections on the floor may add

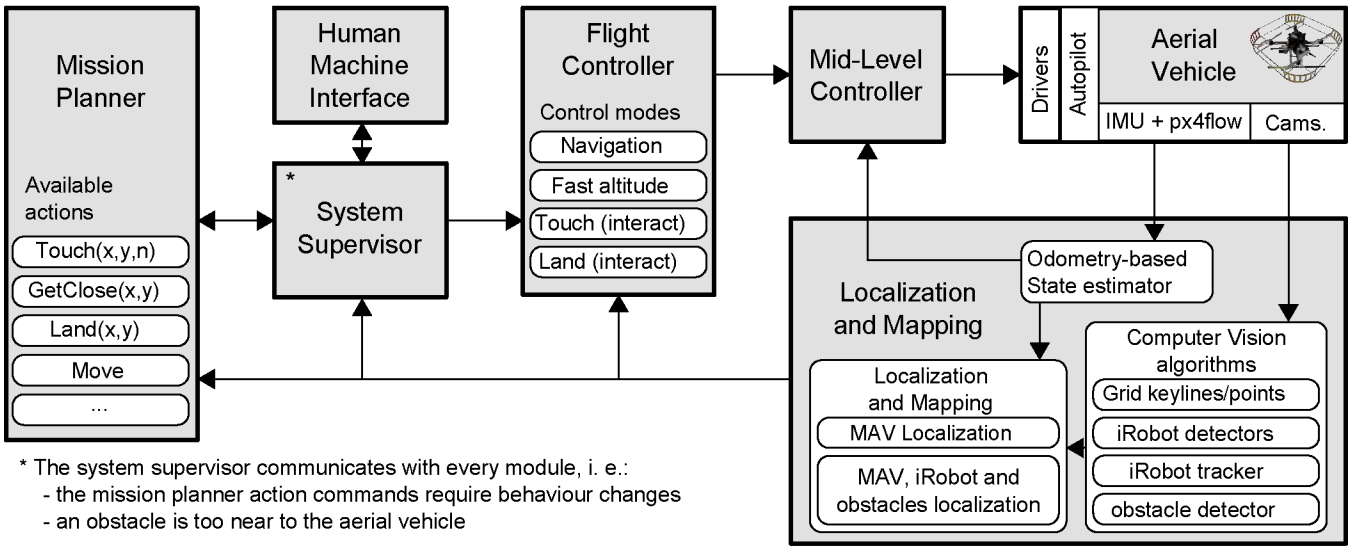


Fig. 3: System Architecture.

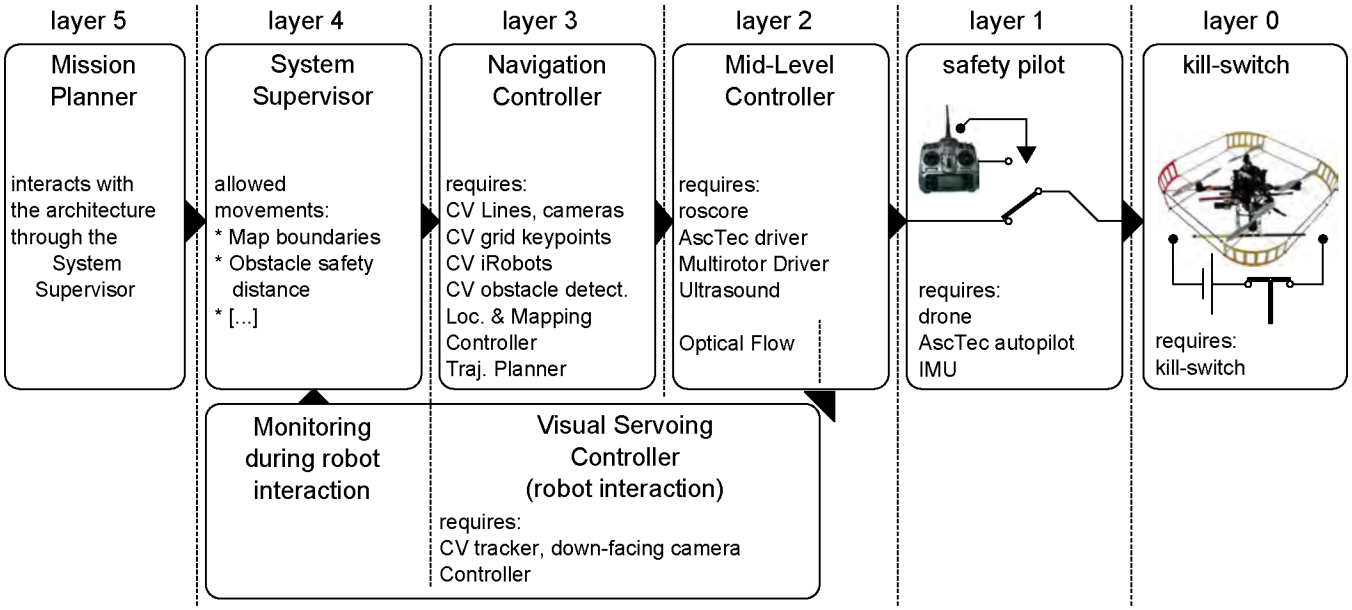
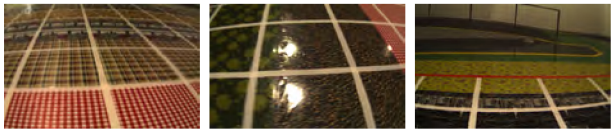


Fig. 4: Safety measures and risk workflow diagram of the proposed architecture.

high luminance regions to the pictures. In addition, at most one key line may appear in the pictures. Examples can be seen in figure 5.



(a) A non-uniform background (b) Light reflections (c) A red key line

Fig. 5: Some pictures of our testing arena taken from our UAV.

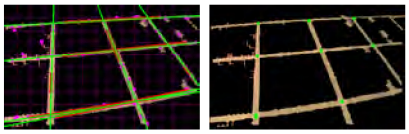
The visual landmarks we seek are: the points where the vertical and horizontal lines of the grid intersect, and the key lines. In order to extract these landmarks, we propose the following algorithm:

1) Segment the picture using an adaptive threshold. As the luminance is the most salient characteristic of the lines, we use the Y channel of the YCrCb colorspace;
2) Refine the segmentation by flood-filling the grid. This enables the removal of unwanted high-luminance blobs;
3) Compute a distance map of the obtained mask. The ridges represent the lines of the grid, and the peaks their intersections. Because of perspective, the farthest lines appear smaller than the closest ones. Hence fixing a threshold cannot extract the peaks; instead, we propose sampling local maxima of subregions of the distance map and fitting lines to these maxima;

4) Divide the distance map into subregions and sample one local maxima by subregion;
5) Use RANSAC [3] to identify possible lines in the picture. Each local maxima can vote for multiple lines;
6) Sort the possible line models obtained by vote count in order to identify the most important ones;
7) Use RANSAC again, using the line models by descending vote order. This time,each local maxima can only vote for a single line, ensuring unicity of the lines.
8) Grid intersections are then computed in a mundane way knowing the equations of the retained lines;
9) Compute the mean color of the neighborhood of each couple of local maxima having generated one of the retained lines in order to identify the key lines.

This process is illustrated in figure 6. In subfigure 6(d), magenta lines and discs respectively show the division of the picture in subregions and the local maxima extracted from each subregion. Green lines are the result of the two-pass RANSAC, and red lines link the two generating points of each green line, which are used for identifying key lines.

(a) Input (b) Threshold mask (c) Floodfilled grid



(d) 2-pass RANSAC (e) Grid intersections

Fig. 6: Illustration of the grid intersection extraction.

In addition to the static components of the arena described in the previous paragraph, iRobots may appear in the pictures. We need to localize them, identify the category they belong to and know the direction in which they are moving in order to provide the information needed for decision-making. The robots are split in two categories:

- *Targets* are robots to be herded; their top plate is red- or green-colored;
- *Threats* are robots carrying poles.

We localize the robots' positions using a Viola-Jones classifier. This classifier has shown in the past excellent real-time results in localizing faces and pedestrians [15], and can be extended to objects [2]. The classifier is speeded-up by using the Run-time Adaptive Sliding Window algorithm [1], which enables the use of different spatial resolutions depending on the output stage of the classifier.

Each detected robot is given an unique identifier in our system when it is detected first. A simple confidence score is also associated to each robot instance:

$$\text{confidence} = \frac{\text{number of successful detections}}{\text{number of frames elapsed since first detection}}$$

A robot instance becomes invalid if the robot is not detected for ten consecutive frames, two detections being considered the same robot if they are not separated by a distance greater than an user-specified threshold. In addition, if an existing robot is not detected in a frame, the Viola-Jones detector is run again in the neighborhood of its last known position with a finer spatial resolution in order not to lose the robots. Our system keeps track of up to ten different robots, which types are identified using color clues : robots with green or red top plates are targets, other colors are considered as threats. An SVM classifier is used to classify the pixels of each detected robot to determine the color of its plate.

Examples of robot and keypoint detections are shown in figure 7. Informations about framerate, number of targets, threats and unknown robots are shown on top of the pictures. The confidence score is shown above the robot and its identifier on its right. The color of the robot's bounding box reflects the one of his top plate. Grid intersections appear as blue circled dots.

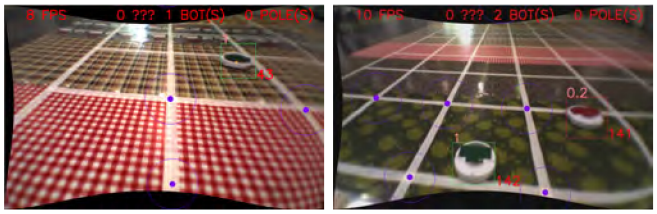


Fig. 7: Robot detection results.

3) Drone Localization and Arena Mapping: This module uses the grid landmarks given by the CV Module and the estimated pose of the drone given by the Odometry State Estimator to create a local map of the Arena and to locate the drone in this map. It can be divided in two submodules: local localization and mapping, and global localization.

The *local localization and mapping* uses the estimated state of the drone given by the state estimator and the grid landmarks obtained by the CV Module. It runs a Simultaneous Localization and Mapping (SLAM) algorithm to generate a local map of the arena composed by the grid landmarks and simultaneously locate the drone in this map. An EKF-SLAM (see [14]) algorithm in the 2D map has been explored as the solution of this part. Figure 8 shows the results of a simulation of this submodule. The map is closed deficiently but errors of less than 1 meter are generated after a circumference exploration trajectory of 9 m radius. This error is due to the correspondences problem because the key points given by the CV Module do not have a unique descriptor. This error is large compared with the size of the drone, yet has little impact as the drone interacts with the target ground robots using algorithms based on their relative pose. Furthermore, the obstacles are localized relatively to a local map, ensuring more close range accuracy.

Other SLAM algorithms that optimize graphs, like the one developed in [5], will be examined in order to improve the results obtained so far, by exploiting the knowledge of a planar grid formed by 1m x 1m squares.

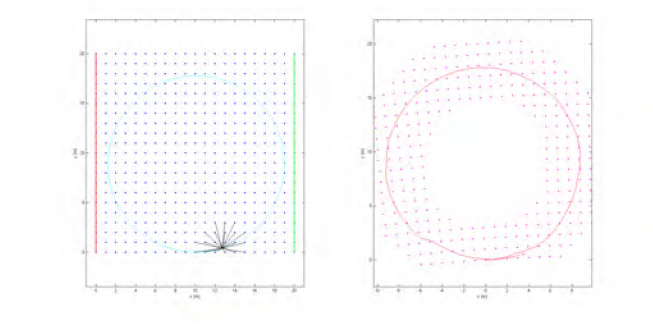


Fig. 8: EKF-SLAM. In the left plot, the Arena grid intersections are represented by blue points. The red and the green line represent the key lines of the Arena. The trajectory followed by the robot is represented in cyan and the grid landmarks given by the CV algorithm at that moment in black. In the right plot, the grid landmarks that have been mapped in the local map are represented in magenta; the estimated trajectory followed by the drone in that local map appears in red.

The *global localization* submodule utilizes the local map, the local drone pose, and the key lines given by the CV module, to locate the drone in the real Arena map. A particle filter including a model of the Arena will be used to achieve this task (see [14]).

The local map is not complete at the beginning of the mission, neither is the global positioning working. Hence the first task to be performed after the take-off and stabilization of the drone is to perform an exploration task aiming at finding a key line.

4) Target Ground Robots and Moving Obstacles Localization: The targets' and obstacles' pose and location are estimated using the information given by the CV module in the form of local coordinates for each individual robot. With this information a local map of the position of the robots is created to estimate their position at each instance and plan the "herding" task according to importance.

To localize and track the robots a Particle Filter (PF) was implemented. In PFs, the key idea is to represent the required posterior density function by a set of randomized samples with associated weights and to compute estimates based on these samples and weights. In order to analyze and make inference about a dynamic system, at least two models are required. First, a model describing the evolution of the state with time (the system model), which models the forward motion of the robot and its turning radius. Second, a model relating the noisy measurements to the state (the measurement model) [14]. Finally, the information of the location of the robots is forwarded to the mission planner.

B. Navigation and Flight Control

The Flight Controller module (see Fig. 3) is in charge of navigating the vehicle, and has four operating modes which prepare the robotic system to execute the action commands sent by the Mission Planner. This module is an extension of the work presented in [12], [11] and [7] and [9].

- Navigation Mode: this mode uses the feedback from the Localization and Mapping modules. It is used to navigate at 2-3 m altitude to desired positions in the map. A obstacle-free trajectory is calculated by a PRM-based trajectory planner.
- Fast Altitude Mode: the altitude is controlled by means of a fast predictive controller. The other degrees of freedom of the vehicle are commanded on open-loop to approximately stay in place. This mode is used when the vehicle is too near the floor where the Localization and Mapping module might fail.
- Two Visual Servoing Modes: these operating modes are used for actions that require the interaction with an iRobot. These modes directly utilize the feedback from the Visual Tracking of the iRobot with which the aerial robot is going to interact. The estimated position of the MAV in the map is unused. However the System Monitor might decide to interrupt the action due to safety concerns, based on feedback from the rest of the architecture.
  - Vertical Touching Mode: the aerial robot is commanded to touch an iRobot from above.
  - Ground-Level Touching Mode: the aerial robot is commanded to land in front of an iRobot.

In addition to the Flight Controller operating modes, the Low-Level Controller has several flying modes: take-off, hover, flying, land and emergency. These flying modes were designed to ease the realization of experiments during testing and also as safety measures. During mission execution, and excluding take-off and landing operations, only the flying mode is utilized.

C. Supervising

The system supervisor is in charge of the following tasks:

- During start-up: it does not allow the software architecture to start the flight until all modules are online. Each module broadcasts a ROS message stating whether the module is started (processing information) or not (idle). The supervisor checks this message to know if any module is not online, or frozen.
- During mission execution: it acts as an interface between the Human Machine Interface (HMI) and the rest of the architecture.
- During mission execution: it supervise the commands sent between the different modules, ensuring that changing the operating modes of the aerial robot is done in the correct step sequences.
- During mission execution: it also monitors some safety measures regarding the rules of the competition. It checks that the vehicle does not got out of the map, and also that there are no obstacles inside a predefined safety distance.

D. Mission Planning

The architecture includes a module for mission planning to generate tactical goals [4]. The mission planner generates long-term goals to achieve in the next seconds. We have designed the mission planner to follow a strategy that we call *search-and-guide*. The UAV first *searches* for an appropriate ground robot considering certain characteristics (distance between UAV and ground robots, distance to the green line, etc.). Then, the UAV *guides* the selected robot to the green line by interacting with the robot at specific moments. During the execution of this strategy, the process can fail because, for example, there are obstacles in the path. In this case, the UAV may decide abandon the selected robot and search for another one. The mission planner works as a deliberative component in our architecture following the hybrid deliberative/reactive paradigm [6]. The mission planner uses as input data the information generated by the perception modules. This includes, for every obstacle, its spatial location and, for every perceived ground robot, its spatial location, speed, orientation and time of cycle (time to the next end of cycle of 20 seconds). The mission planner generates specific goals associated to behaviors that can be understood and achieved by the flight control system. Each reasoning cycle, the mission planner sends a specific goal to the flight controller to be executed. The flight controller tries to achieve the goal and, then, notifies to the mission planner the result i.e., either if the goal has been completed or, on the contrary, it has failed due to certain reasons (presence of obstacles, etc.). Examples of goals are:

- *Search(x, y, n).* The UAV goes to the position *(x, y)* to find at least *n* robots. The UAV can achieve this goal before arriving to the point *(x, y)* if it can observe *n* robots.
- *GetClose(x, y).* The UAV gets close to the ground robot located at the point *(x, y)* up to a prefixed distance (e.g., 1 meter). This movement is limited to a maximum time of *N* seconds (e.g., $N = 5$ seconds).
- *Touch(x, y, n).* The UAV touches *n* times the ground robot that is currently located at the point *(x, y)*.

We have implemented the mission planner as a hierarchical planner. Figure 9 shows a partial view of the hierarchical decomposition of goals and sub-goals. The top level goal, *HerdRobots*, represents the general goal of the UAV, i.e., guide a set of robots to the green line. It can be performed by simpler goals of such as *search(x, y, n)* or *Guide(x, y)*, i.e., guide a particular robot located at *(x, y)* to the green line. In its turn, this second goal can be decomposed into specific goals (get close or touch) or other subgoals. Note that this can be a recursive process, i.e., at the bottom level there can be goals that are also present at upper levels in the hierarchy. For this decomposition process we use production rules. The rules include in their premises qualitative attributes. Table I shows attributes that we use for a ground robot. The values of these attributes are obtained as an abstraction process from the quantitative input data from the observed world. For example, the attribute *DistanceUAV* obtains the qualitative value *far* when the distance between the ground robot and the UAV is greater than 2 meters.

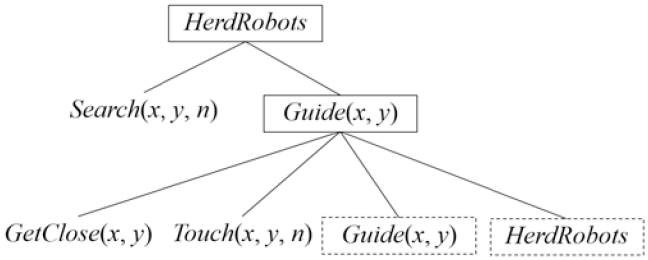


Fig. 9: Example of hierarchy for goal-subgoal decomposition.

Attribute	Values	Description
Out	{*Yes*, *No*}	The ground robot is out of bounds of the arena when the UAV can reach the robot
ReachGreenLine	{*Yes*, *No*}	The ground robot will cross the green line by itself without any interaction with the UAV
NearObstacle	{*Yes*, *No*}	There is at least another robot (ground robot or obstacle) very close to the robot (distance less than 1 m)
DistanceUAV	{*Close*, *Far*}	Distance between the UAV and the ground robot (the distance is far when it is greater than 2 meters)
Direction	{*N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*}	Estimated direction of the robot when the UAV reaches it (represented as cardinal points)
NearReverse	{*Yes*, *No*}	The robot is near the end of the cycle of 20 seconds (when it changes the direction 180 degrees)
FeasibleGuiding	{*Yes*, *No*}	There is enough available time to guide the robot, according to an estimation of the actions to be done

TABLE I: QUALITATIVE ATTRIBUTES OF THE IROBOTS USED BY THE MISSION PLANNER.

The rules also include functions in their premises. For example, the function called *SelectRobot* to select the most appropriate ground robot to be guided (considering, for example, appropriate distances and orientations) or the function *SelectSearchPosition* to select the next point to search ground robots.

E. Human-Machine Interface

During experimental mission executions, the HMI is used mainly for monitoring purposes. It has a terminal console front-end and a visualization front-end based on ROS rviz. The terminal shows a summarized status of each modules, the current action and its execution status, the battery levels and part of the telemetry data. The operator only interacts with the HMI to start the mission by pressing the start key.

During testing, the HMI has a specifically defined set of keys for the current test at hand. It allows to substitute the Mission Planner by an operator, so that experimental tests can be focused on specific behaviors or parts of the architecture.

IV. System Validation

The system has been validated at different levels before the complete architecture is tested in experimental flights. This section explains the utilization of simulations designed to test the Mission Planner, and the State Estimation and Control related modules. Additionally, the Computer Vision algorithms are tested on images taken in a replica of the IARC competition flight field. Finally, in this section, the results achieved by our team during the 2014 competition in the China Venue are shown.

A. Simulation

1) High-Level Simulation: We validated the mission planner with the help of a computer system that simulates the behavior of the UAV and the ground robots. The software components of this simulator, which are divided into three main functional blocks:

- *Scene simulator.* This part of the architecture simulates the movements of the ground robots, the obstacles and the UAV. The ground robots and obstacles are simulated according to the IARC rules.
- *UAV control and perception.* The software architecture includes components that simulate the perception system and the flight control system of the UAV.
- *3D Visualization.* The simulator also includes a component for 3D visualization. This component presents to the user a 3D animated view of the robots and the UAV. The visualization also shows the trajectories followed by the robots, which is useful for the user to analyze the movements. Figure 10 shows an example of the screen presented by the visualization tool.

The Mission Planner was analyzed and evaluated with the help of the simulator to help to select the best strategies.

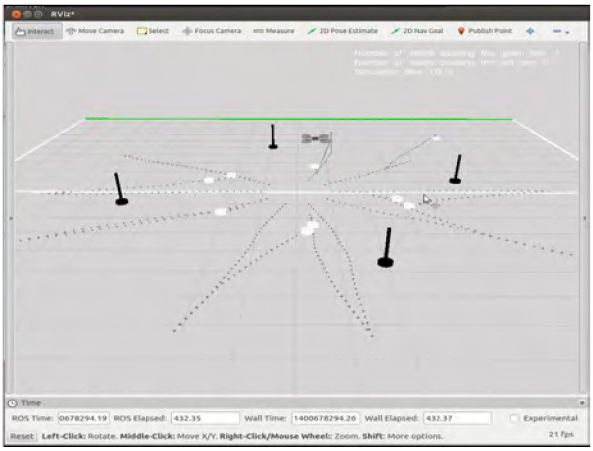


Fig. 10: Example screen presented by the visualization system

2) Mid-Level Simulation: A ROS module that emulates the interface of the flying vehicle and its dynamics has been designed and implemented in order to run tests against the modules under development.

For many of the architecture's modules, the simulator can be run against the modules that will be run during experimental mission execution. This kind of tests can be run by the developer alone and allow to find bugs on the implementation without risking the actual aerial vehicle.

The main drawback of this simulator is the lack of a generation of realistic images that could be processed by the Computer Vision algorithms. Thus, the CV modules must be tested offline separately on data gathered on our replica of the IARC competition field.

B. Field Tests

Field testing is necessary to ensure all components of the drone work well together in a real context. In order to prepare efficiently the drone for the IARC challenge, we have created an arena replica based on the informations given in the challenge rules.

The arena we created following the explanations given in these rules and consists of a grid made of white tape (key lines excepted, for which the tape is colored) over an inhomogeneous background. This background was made of various wallpapers showing different colors and patterns in order to provide harsh conditions for the vision-based algorithms. In addition, we possess five iRobots prepared according to the rules for simulating the mission. Figure 11 shows our arena replica.

Fig. 11: Pictures of the replica arena built for the IARC mission 7 challenge.

Unfortunately, before the competition's day, the full system could not be tested because the computer vision algorithms were giving a poor performance. Nevertheless, partial test were carried out to demonstrate the performance of some modules independently.

C. 2014 Competition Results

We presented our solution to the 2014 competition, in the China venue. We were the only European team participating in this competition. Due to the difficulty of the competition, nobody was able to solve the challenge, neither in the China or in the American venue[1]. That means, the same mission will be repeated next year until any team will be able to finish it.

Our system only partially worked with respect to the planned solution, but we were able to demonstrate the good performance that we have achieved, what reported to us two special awards: Best Target Detection and Best System Control. Some videos shown the performance of our solution

[1] http://www.aerialroboticscompetition.org/stories/stories5.php

and our execution during the competition can be seen in the following links: http://vision4uav.eu/?q=IARC14 and http://vision4uav.eu/?q=IARC14new.

A short explanation of the 2014 competition videos is the following: the fully equipped UAV is autonomously navi- gating in the arena, being able to detect its limits avoiding getting away it. The UAV is also detecting the moving obstacles, planing a trajectory to autonomously navigate in the arena without collisions. The last operative main functionality is the detection of the target robots (although it is not able to touch them). The supervisor, communication and HMI features were as well properly working.

In table II, the main problems that we found that lead us to a lower performance that we expected, are summarized:

V. CONCLUSIONS AND FUTURE WORK

This paper presents the proposed solution and architecture developed by the team from the "Universidad Politecnica de Madrid" to the IARC mission 7 challenge. Among other topics, it addresses the hardware aspects of the aerial robot, a description of each module of the architecture, the testing procedures by simulation and experimental flights, and a study of the safety capabilities of our design.

The design of this architecture has presented several new challenges to our team with regards to our own prior work: it uses Computer Vision in a mostly unstructured environment, the controller requires to switch between various operating modes and to interact with ground robots, and the architecture includes a fully functional Mission Planner which has to take intelligent decisions.

With the experience gained during the 2014 competition, the following milestones are planned to be achieved during the year 2015 (see table II):

- Improve the set-up of the aerial vehicle;
- Improve the software framework;
- Improve the modules used during the interaction with the target ground robots;
- Develop the modules needed for the detection and interaction with other aerial robots;
- Improve the system architecture to allow the drone to move more precisely, at higher speeds and with a better performance while meeting the safety requirements.

All this new improvements and developments will lead us to accomplish the IARC mission 7 challenge.

Problem	Cause	Solution and future solution
Limited payload: magnets and mechanical protections were not mounted onboard the UAV. Only few sensors could be mounted.	The selected UAV was one of the smallest of all the participant UAVs	We reduced the number of sensors by using computer vision and we did not use mechanical protections in all our flights. Neither the magnets for the ground robot interactions were carried. In the future. a more powerful UAV will be used.
Limited HW interfaces	The cameras used had USB 2.0 interfaces. the same as the PX4Flow sensor. All of them used a very high bandwidth	Reduction of the used bandwidth by reducing the framerates of the cameras. In the future. multiple onboard computers will be used. USB 3.0 cameras will be used.
Limited Computer Power	Big amount of data that needed to be processed on a single onboard computer	Reduction of the processed data by reducing the framerates of the cameras (to 10 Hz.). In the future. multiple onboard computers will be used.
Safety problems	The UAV only relies on Computer Vision. so if it fails. the UAV could crash	We implemented some safety modules like the supervisor. In the future. more kind of sensors (like ultrasound) will be used.
Slow and unprecise movement of the UAV	Slow and poor performance of the perception. localization and mapping system (precision between 10 centimeters and 1 meter).	Slow movement to avoid crashes. Use of a very conservative strategy in the trajectory planning. The ground robot interaction was not implemented. In the future. the involved algorithms will be improved by using more sensors and more onboard computers.
Limited environment perception	Cameras allowed us to detect ground robots (target and obstacles) up to 3 meters away from the UAV. The arena perception was also very limited.	The movement of the UAV was slowed down to avoid crashes. In the future. more sensors and better cameras will be used.
Several SW problems like unexpected program crashes	The SW grown more that expected and planned.	A supervisor that monitors the state of all the modules was implemented. In the future. the SW framework will be improved to support a more complex architecture.

TABLE II: Summary of the main problems found during the development of the UAV, the cause, the solution adopted for the 2014 competition and the solution that will be adopted for the following competitions.